Report For Movie Search Engine

Author: Yufei Zhang, Yijun Zhou, Fan Qiao
Github: https://github.com/ffan-web/Movie-Search-Engine

Introduction

Our team has developed a comprehensive movie search engine that is designed to provide users with relevant movie information based on their queries. The primary objective of our search engine is to help users discover new and exciting movies that they might be interested in watching. The basic functionality of our search engine enables users to search for movies using keywords such as movie titles, actor names.

In addition to basic movie search functionality, our system offers movie recommendations based on several factors such as genre, popularity and actors. For instance, if a user searches for a particular movie, our system can recommend popular movies of the same genre, movies featuring the same actors. These recommendations enable users to explore a wide range of movies that they might find interesting.

Our search engine has a user-friendly interface that is easy to use and navigate. The search bar is located prominently on the homepage, enabling users to enter their search queries easily. Moreover, the search results are displayed in a clear and concise manner, making it easy for users to find the movie information they are looking for.

In conclusion, our movie search engine provides a comprehensive search experience. With our advanced search capabilities and recommendation system, we aim to make it easier for users to discover new and exciting movies. Our user-friendly interface and efficient search algorithm make our search engine a valuable tool for anyone looking for movie information.

Prior Work

Adomavicius, G., & Tuzhilin, A. (2005) presents a comprehensive survey of the state-of-the-art in recommender systems, discussing various techniques, algorithms, and evaluation methods. The authors identify key limitations in existing systems and propose possible extensions to enhance recommendation quality and user experience. The paper highlights the importance of incorporating contextual information and developing multidimensional and hybrid recommendation approaches.

Burke, R. (2002) investigates hybrid recommender systems, which combine multiple recommendation techniques to improve performance and overcome individual method limitations. The paper surveys various hybridization approaches and presents experimental results from several hybrid systems. The author demonstrates that hybrid recommender systems can outperform standalone methods by leveraging the strengths of different techniques.

Method

1. Data preparation : We collected data from the IMDB API. This data source provided us with a comprehensive dataset of movie information, including movie titles, full cast, images, types and ratings. The use of this data source ensures that our search engine is equipped with up-to-date and relevant information on movies and actors, enabling us to provide accurate and relevant search results to users.

2. To build a search engine, we utilize Solr, which is a search server built on Lucene that allows for full-text search capabilities. Here are the steps:

    - Install and configure Solr

    The first step in building a search engine using Solr is to install and configure Solr on your machine or server. We downloaded Solr from the Apache Solr website. After we installed Solr, we configured Solr by setting up the Solr core, configuring the schema.xml file to define the fields and their types, and setting up the solrconfig.xml file to define the search settings.

    - Upload data into Solr

    The next step is to upload data into Solr. We have developed a core and named it "myimdb". We use the command-line *./post -c myimdb ../path* to upload our movie data into the Solr.

    - Indexing data

    After uploading data into Solr, the next step is to index the data. Indexing is the process of analyzing the data, creating an index of the data, and storing it in Solr. The Solr will go through the indexing process involves tokenizing the data into words or terms, applying various filters to remove stop words or perform stemming, and then adding the terms to the index.

    - Searching queries

    The final step in building a search engine using Solr is to perform searches on the indexed data. We use queries to search related movie information.

3. Recommendation:

    Our proposed movie search engine operates as follows: Initially, users can search for a movie or actor on the search page. Utilizing Solr, a powerful search platform, the engine returns the top 10 actors and top 10 movies related to the search query. If a user clicks on an actor, the search engine will display information about that actor and recommend movies in which they have appeared. On the other hand, if a user clicks on a movie, the engine will show information about the selected movie and recommend other popular movies within the same genre.

    This approach combines both actor and genre-based recommendations to help users discover a wider range of movies tailored to their preferences. By leveraging Solr, the search engine can efficiently provide accurate and relevant recommendations based on users' interactions with the platform.

Result

1. Result from Solr:

KeyWord Matching

- Movie search : One Day
  Search Query: title: "one day"


```
"response":{"numFound":10,"start":0,"numFoundExact":true,"docs":[
    {
      "id":"tt1297455",
      "image":["https://m.media-amazon.com/images/M/MV5BMTJiYzBiZjktMDMyYS00NjBiLWEyZmItNzNlMzA5MDg3N2Q5XkEyXkFqcGdeQXVyMDkxNTJ
      "title":["One Day"],
      "description":["(2008 TV Movie)"],
      "runtimeStr":["39 min"],
      "genres":["Drama"],
      "genreList":["[{'key': 'Drama', 'value': 'Drama'}]"],
      "imDbRating":[7.4],
      "imDbRatingVotes":[64.0],
      "plot":["Jos is mentally-challenged and still living with his mother. On his 34th birthday his life is about to change d
      "stars":["Jaap van Heusden, Barry Atsma, Kuno Bakker, Leny Breederveld, Benja Bruijning"],
      "starList":["[{'id': 'tt1297455', 'name': 'Jaap van Heusden'}, {'id': 'tt1297455', 'name': 'Barry Atsma'}, {'id': 'tt129
      "Unnamed__0":[7732.0],
      "_version_":1761183895792910339},
```


- Actors search : Leonardo DiCaprio
  Search Query: stars: "Leonardo DiCaprio"


```
{
  "id":"tt9725164",
  "image":["https://m.media-amazon.com/images/M/MV5BNTM5MmUyZDItYmI0Zi00ZjU0LTk2MjQtY2I0Njk4MjkxM2M4XkEyXkFqcGdeQXVyMjM0Mz
  "title":["Leonardo DiCaprio - Phantom und Superstar"],
  "description":["(2015 TV Movie)"],
  "runtimeStr":["55 min"],
  "genres":["Biography"],
  "genreList":["[{'key': 'Biography', 'value': 'Biography'}]"],
  "stars":["Leonardo DiCaprio"],
  "starList":["[{'id': 'tt9725164', 'name': 'Leonardo DiCaprio'}]"],
  "Unnamed__0":[2956.0],
  "_version_":1761183897537740808},
```

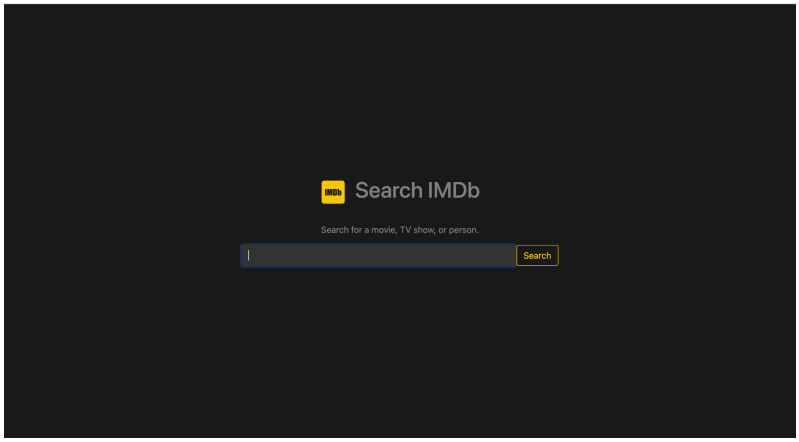

Boolean Query Search

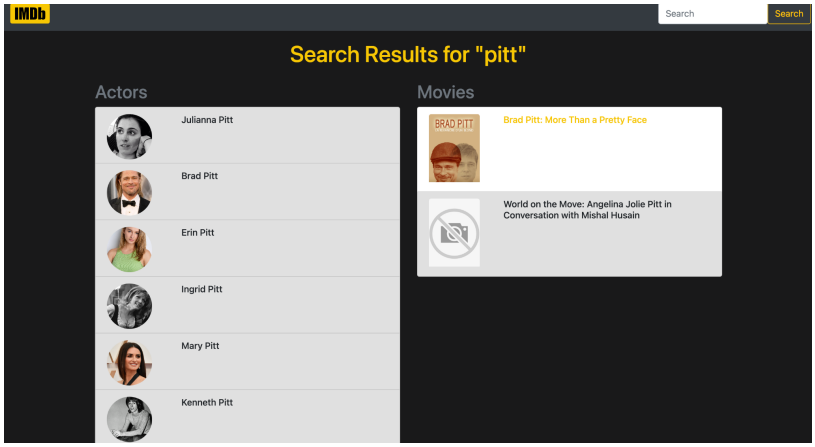
- Movie type and rating score
  Search Query: genres: Comedy AND imDbRating: [8.0 TO *]


```
"response":{"numFound":581,"start":0,"numFoundExact":true,"docs":[
    {
      "id":"tt0304598",
      "image":["https://m.media-amazon.com/images/M/MV5BNDBjOWY2YzAtZGJhZC00NWQ3LTk0YzctOGJkNmJhODFhMTVhXkEyXkFqcGdeQXVyNzg5OT
      "title":["Pyrotehnimata sto tsepaki mou"],
      "description":["(2000 TV Movie)"],
      "runtimeStr":["120 min"],
      "genres":["Comedy"],
      "genreList":["[{'key': 'Comedy', 'value': 'Comedy'}]"],
      "imDbRating":[8.0],
      "imDbRatingVotes":[9.0],
      "stars":["Vasilis Tsivilikas, Kalliopi Evagellidi, Pavlos Haikalis, Panos Hatzikoutselis"],
      "starList":["[{'id': 'tt0304598', 'name': 'Vasilis Tsivilikas'}, {'id': 'tt0304598', 'name': 'Vasilis Tsivilikas'}, {'id
      "_version_":1761183893503868931},
```

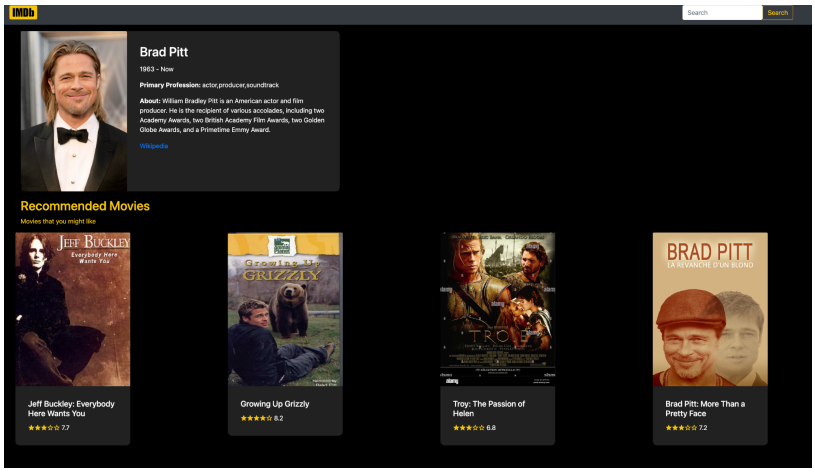
2. Search Result form builted website

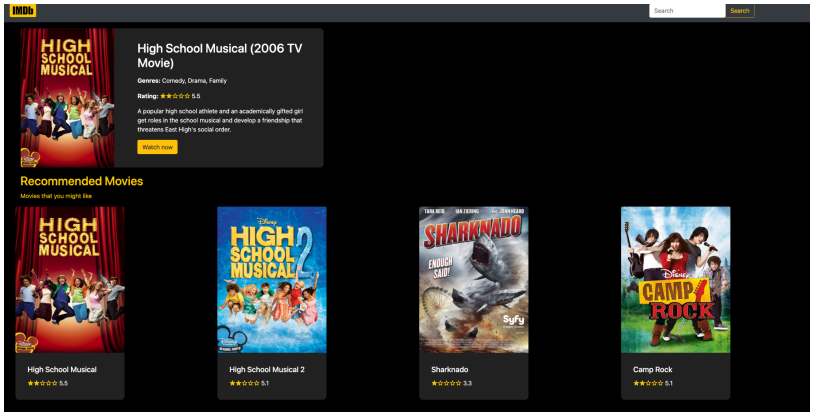
    1) Movie and actors search





    2) Recommendation result





Reference

1. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.
2. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.